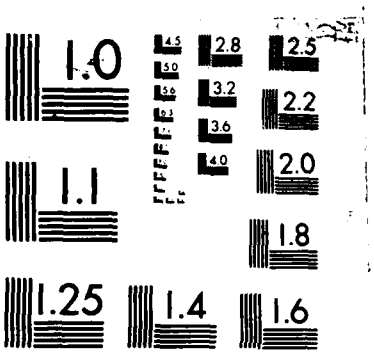
AD-A177 561 PARALLEL LOGIC PROGRAMMING AND ZMOB AND PARALLEL SYSTEMS SOFTWARE AND HAR (U) MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE J MINKER ET AL 1/1

UNCLASSIFIED 30 DEC 85 AFOSR-TR-87-0212 AFOSR-82-0303 F/G 9/2 NL



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

(2)

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 87-0212
6a. NAME OF PERFORMING ORGANIZATION: University of Maryland; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: Air Force Office of Scientific Research
6c. ADDRESS (City, State and ZIP Code)	7b. ADDRESS (City, State and ZIP Code): Directorate of Mathematical & Information Sciences, Bolling AFB DC 20332
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-82-0303
8c. ADDRESS (City, State and ZIP Code): Bolling AFB DC 20332	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
		A7	

AD-A177 561

11. TITLE (Include Security Classification): PARALLEL LOGIC PROGRAMMING AND ZMOB AND PARALLEL SYSTEMS SOFTWARE AND HARDWARE

12. PERSONAL AUTHOR(S): Professor Jack Minker

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 10/10/84 TO 11/1/85	1985, Dec 30	9

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB GR	

DTIC ELECTE MAR 0 3 1987 S D

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The initial version of PRISM uses a simulation of the ZMOB hardware, and has been fully tested and debugged. In addition, several enhancements were made to PRISM to permit experimental analyses to be made, and to incorporate additional features to take full advantage of parallelism in a problem solving environment. Tracing and a statistical gathering package were added to permit experimental analysis. An AND-parallelism capability was added to achieve a second version of the PRISM system, and other features were added to the system to more fully exploit parallelism. Preliminary application and evaluation studies were performed.

FILE COPY

DTIC

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Dr. Robert N. Buchal	(202 767- 4939	NM

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

In the area of systems hardware and software, the ZMOB processor is now fully functional and in everyday use with 128 processors. Work is continuing on an experimental upgrade of some of ZMOB's processors to 68000s. Basic system software for multiprocessing on ZMOB is becoming more robust and performance studies now pinpoint areas for improvement. Studies of parallel software debugging continue to prove the value of multiple program views, and in particular the dicing approach was verified in a controlled experiment. We have also constructed an interactive visual slicer. Studies of the automatic parallelization of programs continues. We can now slice/splice arbitrarily structured programs and have techniques that significantly reduce information overhead between the slices and splicer.

Parallel Logic Programming and ZMOB

and

Parallel Systems Software and Hardware

Annual Report

December 1985

by

Professor Jack Minker

and

Assoc. Professor Mark Weiser

Department of Computer Science

University of Maryland

College Park, Maryland 20742

Abstract

The initial version of PRISM uses a simulation of the ZMOB hardware, and has been fully tested and debugged. In addition, several enhancements were made to PRISM to permit experimental analyses to be made, and to incorporate additional features to take full advantage of parallelism in a problem solving environment. Tracing and a statistical gathering package were added to permit experimental analysis. An AND-parallelism capability was added to achieve a second version of the PRISM system, and other features were added to the system to more fully exploit parallelism. Preliminary application and evaluation studies were performed.

In the area of systems hardware and software, the ZMOB processor is now fully functional and in everyday use with 128 processors. Work is continuing on an experimental upgrade of some of ZMOB's processors to 68000s. Basic system software for multiprocessing on ZMOB is becoming more robust and performance studies now pinpoint areas for improvement. Studies of parallel software debugging continue to prove the value of multiple program views, and in particular the dicing approach was verified in a controlled experiment. We have also constructed an interactive visual slicer. Studies of the automatic parallelization of programs continues. We can now slice/splice arbitrarily structured programs and have techniques that significantly reduce information overhead between the slices and splicer.

TABLE OF CONTENTS

1. Introduction

The purpose of this research is to continue investigate parallel systems software and hardware, and parallel problem solving, under Air Force Grant AFOSR-82-0303.

In the area of parallel problem solving, the initial PRISM has been fully implemented; individual programs have been implemented and tested in a parallel environment; and investigations have been made into extensions to the initial design. In the area of parallel systems hardware and software, the ZMOB is fully functional with 64 processors and in daily use; systems software is reliable; and several advances have been made in the areas of automatically parallelizing programs and understanding their debugging. As a consequence of the work, two book chapters on Zmob have been solicited, one journal article has appeared and two more are in process, several papers were accepted for publication in refereed conference proceedings, three PhD theses, three MS scholarly papers, and over ten technical reports have been written.

The research in parallel problem solving is under the direction of Professor Jack Minker. Dr. Don Perlis whose research has been in non-monotonic logics and artificial intelligence worked with Dr. Minker on some of the theoretical issues associated with non-monotonic logic and parallel logic programs.

The parallel hardware and software effort is under the direction of Associate Professor Mark Weiser. Dr. Weiser was on sabbatical during a portion of this year's period, but was frequently on the Maryland campus and continued to direct the parallel hardware and software research during this time.

This report is subdivided into two major parts. Section 2 describes the accomplished research with respect to PRISM - the parallel problem solving system. Section 3 describes the efforts for the development of systems software and hardware to make the ZMOB system available as a resource for experimentation with parallel algorithms.

2. PRISM and Parallel Problem Solving on ZMOB

There were eight major tasks in parallel problem solving undertaken under the current grant. These are:

(a) Implement PRISM

(b) Test and Debug

(c) Enhancements to Initial System

(d) Application Studies

(e) Evaluation Studies

(f) Control Structure Investigations

(g) Non-Monotonic Logic Investigations

(h) Preliminary Design of PRISM-2.

The first four tasks were to be accomplished under the current grant. There was a considerable amount of testing and debugging accomplished with the PRISM system as anticipated on the simulated belt. A number of application studies were implemented on PRISM to test the program that runs in the VAX.

2.1. Implement PRISM

We review the status of the PRISM implementation effort in the following sections. We describe the status of the major portions of the system and their status.

2.1.1. Design of the Initial Parallel Logic Problem Solver

The functional specification of a system termed PRISM (*P*arallel *I*nference *S*ystem) was essentially completed before the start of the effort. A document termed "Functional Specifications of the ZMOB

Problem Solver" was written and completed during the period of the grant and provides the complete functional specification of the system.

The detailed design of the functional specification is completed. Some modifications have been made as the system progressed.

2.1.2. Implementation

Substantial progress has been made on the PRISM implementation effort. The status of the major parts of the design is as follows.

(1) *User interface with VAX.*
The program provides the interface between the user and VAX, the host computer for PRISM. The program has been implemented and tested. We anticipate no problems of any consequence with the host program. Usage of the host with the system may require some modifications. These cannot be anticipated at this time and the host is essentially completed.

(2) *Extensional Database (EDB) for Moblets.*
The Extensional Database (EDB) is to reside on moblets. An assembly language program was implemented and ran on a single Z80A. When the entire relational table resides on a single moblet, the EDB code responded to queries. The EDB program was loaded on a ZMOB moblet, and tests were made using the belt. The EDB on ZMOB has been run together with the rest of the system with a simulated belt on the VAX. A full test will require the availability of ZMOB. We anticipate that the code for running the EDB with a relational table that exceeds the size of a single moblet will have to undergo extensive testing and possible reworking.

Since the ZMOB was not available, it was decided to implement the EDB in C to permit it to be run in the VAX on the simulated ZMOB belt. The programs to perform this have been designed and implemented on the VAX.

(3) *Extensional Database for VAX.*
The EDB is constructed by the user in the VAX machine. The database supplied by the user is stored initially in the VAX. Syntax checks are performed on the input data, and errors are reported to the user. The data is then indexed, and formatted for transmission to the appropriate number of moblets required. Implementation of this portion of the system is complete. The functions of this program were extended to handle the user specification of the configuration.

(4) *Intensional Database (IDB) for ZMOB and Monitor.*
The Intensional Database (IDB) stores the procedures in the system. The initial design does not permit the number of procedures to exceed the size of a moblet. The IDB including predicates with functions as arguments has been tested on the VAX and is now operational. The IDB monitor has been implemented.

(5) *Intensional Database (IDB) for VAX.*
As with the EDB, the IDB is entered into the VAX by the user. The program does syntax checks on the input data, coordinates with EDB data provided by the user, indexes the procedures and prepares the programs and data for transmission to the moblets. The program has been implemented and tested. It has been interfaced with the user interface program, and the Problem Solving Machine.

(6) *Problem Solving Machine (PSM).*
The Problem Solving Machine is the central portion of PRISM. The initial version of the PSM program has been implemented in C. The PSM has been tested together with the host interface and the IDB on a simulated belt on the VAX. The ability to handle evaluable predicates has been incorporated into the system. Conversion to moblets will not be possible without a major redesign of the PSM. An optimizing compiler that transforms C code for the VAX to moblets will be required. Even if such a compiler becomes available, it is doubtful that the PSM code can fit in a 64 K byte moblet.

(7) *Communication Primitives*
The communication primitives have been implemented and require ZMOB for a full test.

(8) Simulated ZMOB Belt
To further the implementation, a portion of ZMOB is simulated on the VAX and interfaces with the existing programs. The system consisting of the user-VAX interface, the IDB programs, the EDB programs and the PSM, has been tested on a simulated ZMOB belt . This has allowed us to test several portions of the system together rather than in isolation.

(9) *PRISM C Compiler Preprocessor*
A PRISM C preprocessor was developed in order to maintain compatibility between the PRISM C code on the Unix systems and for the ZMOB. The Vandata C cross-compiler, which is being used by the ZMOB project is not fully compatible with the Berkeley 4.2 C compiler which was used during the development of the PRISM software. This became apparent during the development of PRISM for ZMOB. The use of the preprocessor frees one from undesired inconsistencies of the cross compiler.

The differences between the two compilers are three-fold. First, there are no enumerated types in the cross-compiler. Second, it only uses the first eight characters of an identifier. (Externally defined identifiers are further restricted to seven characters). Lastly, it has no structure assignments.

Given an error free C program, the PRISM C preprocessor outputs a modified C program that is functionally equivalent to the original. For each of the above mentioned constructs, the input code is changed to an equivalent construct recognized by the cross-compiler. Although an effort was made to make the output of this preprocessor readable for debugging purposes, one should think of its processing as simply another pass of some C compilation process.

(10) *Modifications to PRISM to Run Under Unix 4.2*
The primary development of PRISM has taken place on a VAX under Berkeley Unix. This was because no software tools for the ZMOB, other than a compiler, were available at the time this implementation was begun. Also, this implementation allowed us to focus on the implementation issues of parallel logic programming without burdening ourselves with the limitations of the Z-80 system.

This year the department changed from the 4.1 release of the Berkeley software to the 4.2 release. Several new primary computing machines were added, a Pyramid 90x being one of them.

In order to have the PRISM system available on all our primary machines it was ported over to the Pyramid as well as modified to run under the 4.2 release. The inter-process communication (ipc) supported by 4.2 is completely different from that supported by 4.1. Since the PRISM system is implemented as a set of communicating processes with one process per physical ZMOB processor and a ZMOB belt simulator process, it makes heavy use of ipc. Thus the conversion from 4.1 to 4.2 was non-trivial and required modification of a large amount of communication code primarily in the PRISM library routines and an almost complete reworking of the belt simulator. This conversion process extended over several months and was completed in early 1985.

2.2. Test and Debug

The system as implemented has undergone substantial testing and debugging. In particular, the communication portion of the system has been tested and is responding as designed. Messages that are sent to form new PSMs, and to make requests of IDBs and EDBs are received and acted upon appropriately. Response messages are sent and received by the PSM. The entire PRISM system has undergone integration testing and works appropriately.

A useable ZMOB system was made available for use towards the beginning of 1985. Unfortunately the size of the moblets, 64K bytes, and the size of the PSM code precludes the possibility of moving the system to ZMOB.

2.3. Enhancements to Initial System

To be an effective research tool enhancements are required to the initial system. Two major enhancements were required:

(a) Tracing and

(b) Statistics

A tracing capability, required to permit the user the ability to monitor the search process was implemented. A trace of the search steps is maintained in the VAX machine and permits the user to follow the proof process as the search progressed and to obtain a printout at termination. The design of the initial system has indicated additional features that are required for the system.

Statistical data will be necessary from each portion of the system so that an analysis can be made of where bottlenecks and problems arise . A preliminary functional specification has been made of the statistical information required.

The above two enhancements were required to permit applications to be tested and run on the system, and to be able to analyze the results.

A revised version of PRISM is being implemented as of the writing of this proposal to include first, a limited version of AND-parallelism resticted to handling independent subproblems; and second, a full version of AND-parallelism. The full version of AND-parallelism will be integrated into the system once the first version is tested successfully.

2.4. Application Studies

The effectiveness of PRISM will, to some extent, be determined by its ability to provide a natural and an efficient environment for a large class of application programs.

Several problems which exercise the important features of PRISM have been written in the PRISM language and tested on the PRISM system. Some of the problems tested and the categories of these problems are :

(1). Database Problems. A large database consisting of information on schools has been tested on PRISM. Other database problems have also been run on the system

(2). Knowledge Base Systems. A knowledge base system for "kinship relationships in a small community" has been implemented on PRISM. Features like AND-selection control and OR selection control have been used to control the search paths. The use of such facilities by novice users to improve the performance of their programs are being studied. The effect of changing the system configuration and the effect of re-deployment of a configuration are also being studied. Other programs like an expert system for medical diagnosis and a MYCIN-like system for bacteria identification are being considered.

(3). Simulation. A chemical synthesis program has been run on PRISM.

(4). Games. Programs for the multiple queens problem, a chess program for end game study of "mate-in-two" combinations have also been implemented.

(5). Miscellaneous. Some programs in the world of robots, tree ordering programs, natural language parsing, and other programs have also been written in the PRISM language and tested.

2.5. Evaluation Studies

When the statistic gathering programs have been integrated with the system, we expect to rerun all of the above programs to gather data on the effectiveness of the language, configuration and parallellism in PRISM. We expect to investigate areas other than those listed above, convert the problems to PRISM logic programs and test them. The results of the tests must be analyzed, and the system performance

assessed.

2.6. Control Structure Investigations

One of the major features of PROLOG is the simplicity of its control structure. It uses a left-to-right selection strategy of atoms in a clause and does a depth-first search. The simplicity of its control structure makes implementations relatively straight forward, and in a sequential system the time to make a decision on what to do next is very rapid.

In artificial intelligence applications it is necessary to have a more elaborate control structure if contextual information is to be used to control the search, and if one wants to achieve a truly parallel system. Alternative approaches to achieving a flexible control structure have been outlined in Kohli [1986]. He describes the advantages and disadvantages of the alternative approaches. We envision a family of different systems in which the user specifies the control features that he desires. For efficiency, these control statements are then compiled and a new system, tailored to the particular application can be achieved. We plan to investigate the details of how such a generator can be developed so as to retain an efficient system. These investigations are planned in the context of a parallel problem solving system.

2.7. Non-Monotonic Logic Investigations

In the previous grant we obtained completeness and soundness results for a class of problems in protected circumscription. We continued our work in non-monotonic logic both in the theoretical aspects and in the practical aspects.

In the theoretical part we investigated McCarthy's notion of formula circumscription (McCarthy [1984]) and found a partial completeness result for formula circumscription. We also investigated computational issues associated with protected circumscription and developed an algorithm that extends the concept of relational databases and permits complete and sound answers to be found in the case where the theory consists of ground atomic formulae extended to contain protected data. We then investigated the general case of Horn deductive databases augmented with protected data and found that the obvious extension of our algorithm to this case was not adequate.

2.8. Preliminary Design of PRISM-2

In the design of PRISM-1 we made some simplifying assumptions, such as allowing AND-parallelism only in the case of independent sub-problems in goals, the IDB must be of sufficient size to remain in a single moblet (although we allow there to be multiple copies of the IDB), we do not permit the user to enter new data into PRISM, but must do so off-line, we do not permit integrity checks to be made on the data entering the database, and we do not have a fully flexible control structure. Once we obtain some experience in how PRISM runs on the ZMOB, we will be in a position to review the design with the objective of making it more flexible.

The field of parallel computation of programs is in its infancy. The experience we gain with PRISM-1 will permit us not only to develop a better system, but will permit us to envision architectures that can be developed to enhance parallel computation. Our work is, perhaps most closely related to the work in Japan on the Fifth Generation Systems.

2. Parallel Hardware and Software

The parallel hardware and software research under this grant has the following goals:

(a) Construction and evaluation of the Zmob architecture.
(b) Construction and evaluation of a variety of systems software for Zmob.
(c) Investigation of languages and operating systems for parallel computation.
(d) Investigation of general strategies for parallel computation.

Work has been proceeding in all four areas, although (a) and (b) have necessarily dominated the early phases of research. There has been good progress, and research in future years will focus on evaluating Zmob and investigating more general concerns of parallelism as guided by the Zmob experience.

2.1. Construction and evaluation of Zmob

At the time of our last report the ZMOB was running with only 16 processors and not reliably at that. It is now running with a full 128 processors, is in daily use for a variety of research projects (primarily systems and numerical analysis).

The experience of making ZMOB has been extremely fruitful for the maturing of parallel architecture research at Maryland, and we are looking forward to the opportunity to design and construct the TWO-MOB over the next 3 years. Most of the problems with getting ZMOB running were simply hardware inexperience, primarily in the areas of power-supplies and transmission line noise. The Computer Science Department has now made a committment to two staff Electronics Engineer lines, and it is the persons on these lines who have made the difference in bringing ZMOB to fruition in the past year.

The 128 processor ZMOB operates at the full clock rate of 10 Mhz, up from the 6.5Mhz of the early ZMOB's. The slower clock had forced software to use the slower programmed I/O on the Z-80A rather than the much faster DMA on the ZMOB processor boards. Thus the faster clock speeds up parallel communication by even better than the apparent 1.5 factor.

Among the special hardware purchased to support ZMOB research this were several disk drives, two high-speed modems and a printer. The disk drives so far have not been integrated into the general use of ZMOB, mostly for hardware reasons. This should happen this year. The modems and printer were purchased to ease access for ZMOB researchers, and have proven quite useful.

We have been doing significant planning for replacing several of the ZMOB Z-80A processor boards with Motorola 68000 boards. This field is changing rapidly, and we have been waiting in part for prices and features to stabilize. We have been working closely with a local manufacturor of these boards who is interested in making use of some of our software and hardware expertise, and we have requisitioned for sixteen 68010 boards, each with 1 Mbyte RAM, DMA, and serial and parallel interfaces. A major advantage of these boards is that PRISM will be able to run on the real ZMOB rather than in simulation on a vax.

As ZMOB has begun to work there has been a renewed interest in its architecture. A recent invited conference paper (Weiser et al [1985]) and two planned invited book chapters on ZMOB attest to this.

2.2. Construction and evaluation of systems software

Any new architecture needs considerable attention to its underlying support software before it can be used. This support software includes debuggers, cross-compilers, assemblers, 'tweakers', linkers, loaders, simulators, etc. This software exists for ZMOB in abundance, and is in a fairly stable state. In fact, the majority of it achieved a milestone this year by moving from research status and hence the responsibility of this grant to 'supported' status and hence a standard part of the Maryland CS Department Laboraratory software, supported by the systems staff. Previous proposals and reports have dwelt on this systems software, but will do so no longer. No more research: it is there, use it.

The evaluation of this systems software is now well underway. There are several projects to measure the overhead of communicating and programming on ZMOB, the net result of which at the moment is that one is better off in assembler language with the current ZMOB. The main reason is the mismatch of the C language with the Z-80 and ZMOB architecture, which is due in part to the lack of DMA at the current clock rate. We have also measured, using the current software, essentially linear speed-up in certain numerical analysis problems as the number of processors goes from 1 to 63. This is an encouraging, if problem dependent, result. We certainly plan more work along these lines.

Another important systems event was the development by us of support in the Unix 4.2bsd kernel for the Xerox XNS protocol suite. This is important for investigating parallelism over networks other than ZMOB and for comparison with ZMOB. This implementation was reported (O'Toole et al [1985]) and since then has been distributed to several other universities.

2.3. Investigation of general strategies for parallelism

We have several theoretical results in the area of general strategies for parallelism. In previous years we developed a theory of program slicing which describes how to produce program subsets which model projections of a program's behavior. These subsets could be run in parallel, and in principle

contained enough information to reconstruct their original program's behavior. The proof of this principle required the theory of 'splicing', in which it was shown that for structured programs this behavior reconstruction could be done in real-time.

The recent work of Mark Weiser's graduate student Mike Mazurek extends these ideas in several ways. First, it presents splicing in graph theoretic terms. Second, it extends the splicing theory to general programs and their graphs , not just structured programs. Third, if one is interested in a projection of the original program's behavior, say just the relative ordering of the output statements, this work presents a technique that significantly reduces the amount of information that has to be spliced. Finally, it has definitions and proofs which are much more elegant than the original language-theoretic approach. The paper "Towards the Automatic Parallelization of Sequential Programs" is in preparation. We expect it will be of interest in the field of parallel programs as well as in the fields of graph theory and abstract automata, to which the results generalize.

A long standing problem in slices is handling program input. Earlier work requires that all input data be sent to all slices, with the possibility that the data is simply ignored by most of them. This carries the communication cost of moving that data around, and the slice run-time cost of reading it in, possibly in a loop with its associated overhead. We refer to this problem as the "splitting" problem. We now have a technique that enables a slice to "bypass" the input of an input statement, if it cannot possibly affect (as detectable by data flow analysis) its computation.

3. Bibliography and References

(1) Cao, D., "Design of the Intensional Database System ofthe ZMOB Parallel Problem Solver", Technical Report, Computer Science Department, University of Maryland, 1982.

(2) Chakravarthy, U.S., "Semantic Query Optimization", Ph. D. Thesis, Department of Computer Science, University of Maryland, 1985.

(3) Chakravarthy, U.S., Minker, J. and Tran, D., "Interfacing Predicate Logic Languages and Relational Databases". *Proceedings of the First International Logic Programming Conference*, September 14-17, 1982, Faculte des Sciences de Luminy Marseille, France, 91-98.

(4) Eisinger, N., Kasif, S., and Minker, J., "Logic Programming: A Parallel Approach", *Proceedings of the First International Logic Programming Conference*, September 14-17, 1982, September 14-17,1982, Faculte des Sciences de Luminy Marseille, France, 71-77.

(5) Eisinger, N., Kasif, S., and Minker, J., "Logic Programming: A Parallel Approach". Technial Report TR-1124, Computer Science Department, University of Maryland, December 1981.

(6) Gal, A. and Minker, J. "A Natural Language Database Interface Giving Cooperative Answers", Department of Computer Science, University of Maryland, 1985.

(7) Gallaire, H., Minker J. and Nicolas, J-M., "Logic and Databases: A Deductive Approach", Computing Surveys, Vol. 16, No. 2, pp. 153-185, June 1984.

(8) Gallaire, H., Minker, J. and Nicolas, J.-M., "Advances in Data Base Theory, Volume 2", Plenum Publishing Company, February, 1984.

(9) Kasif, S. and Minker, J. "The Intelligent Channel: A Scheme for Result Sharing in Parallel Logic Programs", Proceedings International Joint Conference on Artificial Intelligence, August 1985.

(10) Kasif, S., "Analysis of Parallelism in Logic Programs", Ph. D. Thesis, Department of Computer Science, University of Maryland, December 1985.

(11) Kasif, S., Kohli, M., amd Minker, J., PRISM: A Parallel Inference System for Problem Solving, Proceedings of the 8th International Joint Conference on Artificial Intelligence, 8-12 August 1983, Karlsruhe, West Germany, 544-546.

(12) Kasif, S., Kohli, M., and Minker, J., "PRISM - A Parallel Inference System Based on Logic", Technical Report TR-1243, Computer Science Department, University of Maryland, February 1983.

(13) Kasif, S., Kohli, M., and Minker, J., "PRISM: A Parallel Inference System for Problem Solving", Proceedings Logic Programming Workshops, Praiada Falesia, Algarve, Portugal, Universidad Nova De Lisboa, 26 June - 1 July 1983, 123-152.

(14) Kohli, M. "Controlling the Execution of Logic Programs", Proposed Ph. D. Thesis, Department of Computer Science, University of Maryland, May 1986.

(15) Kohli, M., and Minker J., "Control of Logic Programs Using Integrity Constraints" Proceedings Logic Programming Workshop 1983, 26 June - 1 July 1983, 123-152.

(16) Kohli, M., and Minker, J., "A Theory of Intelligent Forward and Backward Tracking", Technical Report (in preparation), Computer Science Department, University of Maryland, March 1983.

(17) Kohli, M., and Minker, J., "Intelligent Control Using Integrity Constraints", Proceedings of the National Conference on Artificial Intelligence, August 22-26, 1983, 202-205.

(18) Lyle, J. Evaluating Variations on Program Slicing for Debugging. Ph.D Dissertation, Computer Science Dept., University of Maryland. December 1984.

(19) Lyle, J. and Weiser, M. Evaluating variations on program slicing for debugging. in preparation, 1985.

(20) Minker, J. Perlis, D. and "Completeness Results for Circumscription", Department of Computer Science, University of Maryland, January 1985.

(21) Minker, J. and Perlis D., "Applications of Protection of Circumscription", Proceedings of the Conference on Automated Deduction- , Napa, California, March 1984.

(22) Minker, J., and Perlis, D., "On The Semantics of Circumscription", Technical Report 1341, Computer Science Department, University of Maryland, August 1983.

(23) Minker, J., et al., "Functional Description of the ZMOB Parallel Problem Solving System", Technical Note 1, Department of Computer Science, University of Maryland, December 1982.

(24) Minker, J., et al.,"Parallel Problem Solving on ZMOB", Proceedings of Trends and Applications 83, Washington, D.C.,1983.

(25) O'Toole, J., Torek, C., and Weiser, M. Implementing XNS protocols for 4.2bsd. Usenix Unix Users Conference, Dallas TX. January 1985.

(26) Weiser, M., Kogge, S., McElvany, M., Pierson, R., Post, R., and Thareja, A. Status and Performance of the Zmob Parallel Processing System. IEEE CompCon conference, San Francisco, CA, February 1985.

END

4-1-87

DTIC